Encryption and Attacks

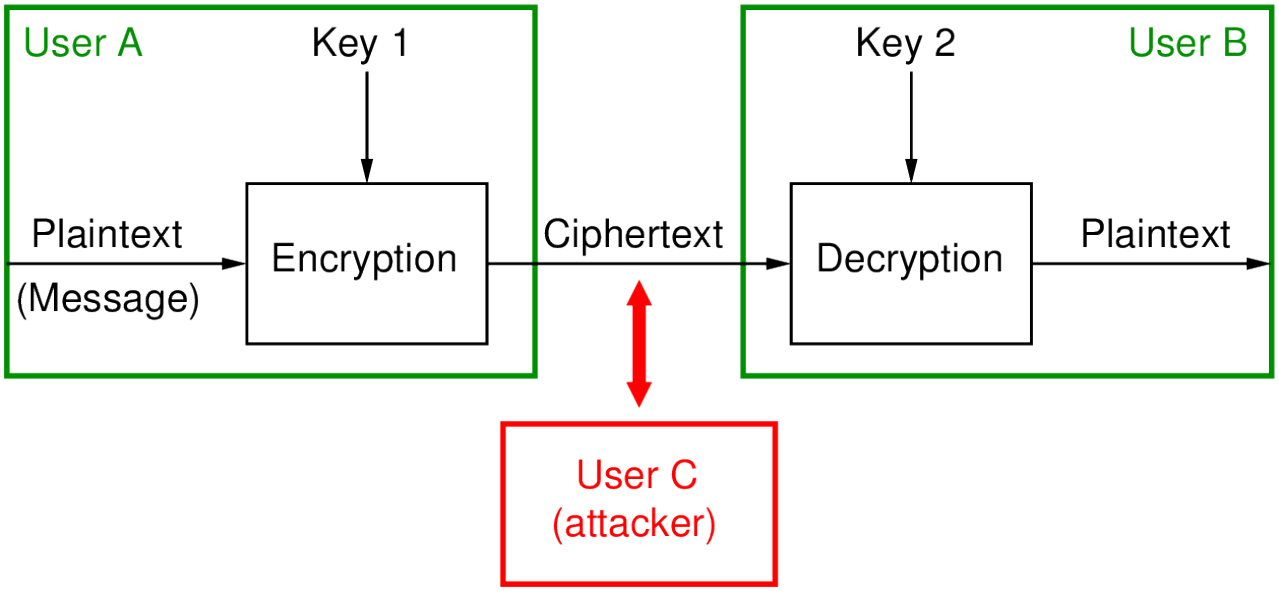
Cryptography

School of Engineering and Technology
CQUniversity Australia

Prepared by Steven Gordon on 04 Jan 2022,
encryption.tex, r1965

Contents

Model of Encryption for Confidentiality



The figure on slide 3 shows the general model for encrypting for confidentiality that we have seen previously.

Characterising Ciphers by Number of Keys

Symmetric sender/receiver use same key (single-key, secret-key, shared-key, conventional)

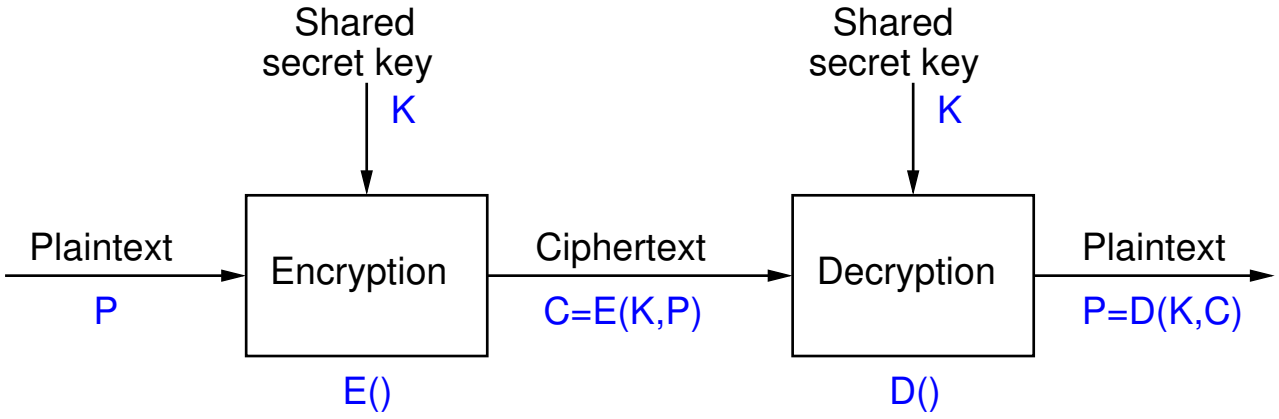
Public-key sender/receiver use different keys (asymmetric)

All ciphers until about the 1960's were symmetric key ciphers. The encrypter and decrypter used the same key, i.e. symmetry between the keys. The key must be shared between the two users and kept secret.

A new form of cryptography was designed in the 1960's and 1970's, where the encrypter uses one key and the decrypter uses a different but related key. The keys are asymmetric. One of the keys is kept secret, while the other can be disclosed, i.e. made public.

We will focus on symmetric key ciphers initially, and return to public-key ciphers later.

Symmetric Key Encryption for Confidentiality



We often use simple mathematical notation to describe the steps. E() is a function that takes two inputs: key K and plaintext P. It returns ciphertext C as output. E() represents the encryption algorithm. D() is the decryption algorithm.

Symmetric key encryption is the oldest form of encryption and involves both parties (e.g. sender and receiver) knowing the same secret key. Plaintext is encrypted with the secret key, and the ciphertext is decrypted with that secret key. If anyone else (i.e attacker) learns the secret key, then the system in not secure.

For symmetric key encryption to be secure, the algorithm must be well designed (strong, not easy to break) and the secret key must be kept secret. AES is an example of a strong algorithm, and it uses keys of length 128 bits or longer. One of the challenges of symmetric key encryption is informing the receiver of the secret key in advance: it must be done in a secure manner.

Common Operations in Symmetric Ciphers

Substitution replace one element in plaintext with another

Permutation re-arrange elements (also called transposition)

Product systems multiple stages of substitutions and permutations, e.g. Feistel network, Substitution Permutation Network (SPN)

Symmetric key ciphers are designed around two basic operations: substitution and permutation. We have seen these operations when looking at classical ciphers. We also saw the principle that repeating the operations can make a cipher more secure. Modern ciphers are designed using these two basic operations, but repeated multiple times. For example, perform a substitution and then permutation, then repeat. The result is a "product system".

The Feistel network and SPN are two common design principles for modern ciphers and will be mentioned later when discussing block ciphers like AES and DES.

Characterising Ciphers by Processing Plaintext

Block cipher process one block of elements at a time, typically 64 or 128 bits

Stream cipher process input elements continuously, e.g. 1 byte at a time, by XOR plaintext with keystream

Originally the idea was that block ciphers were suitable for processing large amounts of data when there were no strict time constraints. Stream ciphers were fast and suitable for real-time applications. For example, for encrypting real-time voice, as the data (plaintext) is generated, it needs to be quickly encrypted and then the ciphertext transmitted across a network. By encrypting only a small amount of plaintext at a time and using the extremely fast XOR operation, stream ciphers could perform the encryption without introducing significant delay.

However nowadays, the dedicated hardware support for block ciphers like AES, there is not a significant difference in performance (delay) of block and stream ciphers. Hence we see block ciphers (in particular, AES) used in scenarios for which stream ciphers were originally designed for.

We will focus on block ciphers initially, and return to stream ciphers later.

Two Important Symmetric Key Block Ciphers

Data Encryption Standard (DES) Became a US government standard in 1977 and widely used for more than 20 years; key is too short

Advanced Encryption Standard (AES) Standardised a replacement of DES in 1998, and now widely used. Highly recommended for use.

While no longer recommended or in widespread use, DES was the first cipher that saw widespread use. The primary limitation of DES however was the key was eventually subject to a brute force attack. It was only 56 bits.

While Triple DES, which used the original DES but expanded the key length, was popular for awhile, a new cipher was needed to perform well in a variety of hardware platforms. AES was standardised in 1998 and continues to be the recommended symmetric key block cipher for most applications today. There are no known practical attacks that cannot be defended.

DES and AES are covered in depth later.

Common Symmetric Key Block Ciphers

Cipher	Year	Designers	Block Size	Key Size	Design
DES	1977	IBM/NSA	64	56	Feistel
IDEA	1991	Lai and Massey	64	128	Other
Blowfish	1993	Schneier	64	32-448	Feistel
RC5	1994	Rivest	64, 128	-2040	Feistel-like
CAST-128	1996	Adams and Tavares	64	40-128	Feistel
Twofish	1998	Schneier et al	128	128, 192, 256	Feistel
Serpent	1998	Anderson et al	128	128, 192, 256	SPN
CAST-256	1998	Adams and Tavares	128	-256	Feistel
RC6	1998	Rivest et al	128	128, 192, 256	Feistel
AES	1998	Rijmen and Daemen	128	128, 192, 256	SPN
3DES	1998	NIST	64	56,112,168	Feistel
Camellia	2000	Mitsubishi/NTT	128	128, 192, 256	Feistel

The figure on slide 9 lists common symmetric key encryption block ciphers starting with DES, through to around the time of AES. Most block ciphers operate on blocks of 64 or 128 bits, and support a range of key lengths. There are three main design principles: Feistel network or structure, Substitution Permutation Network, or Lai-Massey.

AES is still highly recommended for most applications. There have been newer proposals since then, however very few are standards or see wide spread usage. A recent trend is on developing "lightweight" ciphers that perform well on very small devices, e.g. sensors.

A detailed review of block ciphers is Roberto Avanzi's "A Salad of Block Ciphers: The State of the Art in Block Ciphers and their Analysis", 2017, which is available for free at https://eprint.iacr.org/2016/1171.pdf

Contents

Aims and Knowledge of the Attacker

- Study of ciphers and attacks on them is based on assumptions and requirements
  - Assumptions about what attacker knows and can do, e.g. intercept messages, modify messages
  - Requirements of the system/users, e.g. confidentiality, authentication
- Normally assumed attacker knows cipher
  - Keeping internals of algorithms secret is hard
  - Keeping which algorithm used secret is hard
- Attacker also knows the ciphertext
- Attacker has two general approaches
  - "Dumb": try all possible keys, i.e. brute force
  - "Smart": use knowledge of algorithm and ciphertext/plaintext to discover unknown information, i.e. cryptanalysis

Worst Case Brute Force Time for Different Keys

Key length	Key space	Worst case time at speed: 10^9/sec	10^{12}/sec	10^{15}/sec
32	2^{32}	4 sec	4 ms	4 us
56	2^{56}	833 days	20 hrs	72 sec
64	2^{64}	584 yrs	213 days	5 hrs
80	2^{80}	10^7 yrs	10^4 yrs	38 yrs
100	2^{100}	10^{13} yrs	10^{10} yrs	10^7 yrs
128	2^{128}	10^{22} yrs	10^{19} yrs	10^{16} yrs
192	2^{192}	10^{41} yrs	10^{38} yrs	10^{35} yrs
256	2^{256}	10^{60} yrs	10^{57} yrs	10^{54} yrs
26!	2^{88}	10^{10} yrs	10^7 yrs	10^4 yrs

The table on slide 12 shows, for different key lengths, the time it takes to try every key if a single computer could make attempts at one of three rates: 10^9 per second, 10^{12} per second, or 10^{15} per second. There are not necessarily realistic speeds, although roughly represent lower and upper limits for today's computing power.

While this table presents the worst case time, in most cases, it is not much different from the average time. Recall the average time is about half of the worst case time. For a 128 bit key at 10^{15} decrypts per second, the worst case time is about 1×10^{16} years, and the average time is about 0.5×10^{16}. That is, both about 10^{16} years. With such large times, cutting the time in half makes no practical difference.

Note that the last line is for a key for a monoalphabetic English cipher. There are 26! possible keys which is equivalent to a binary key of about 88 bits.

For comparison, the age of the Earth is approximately 4×10^9 years and the age of the universe is approximately 1.3×10^{10} years.

Classifying Attacks Based Upon Information Known

1. Ciphertext Only Attack
2. Known Plaintext Attack
3. Chosen Plaintext Attack
4. Chosen Ciphertext Attack
5. Chosen Text Attack

We describe the different attacks in the following.

Ciphertext Only Attack

- Attacker knows:
  - encryption algorithm
  - ciphertext
- Hardest type of attack
- If cipher can be defeated by this, then cipher is weakest

The common assumption is that an attacker knows the encryption algorithm and ciphertext, and that they had no influence over the choice of ciphertext. This is referred to a *ciphertext only* attack. A cipher that is subject to a ciphertext only attack is the weakest of the groups of attacks we will consider.

Known Plaintext Attack

- Attacker knows:
  - encryption algorithm
  - ciphertext
  - one or more plaintext–ciphertext pairs formed with the secret key
- E.g. attacker has intercept past ciphertext *and* somehow discovered their corresponding plaintext
- All pairs encrypted with the same secret key (which is unknown to attacker)

In a KPA, the attacker also has access to one or more pairs of plaintext/ciphertext. That is, assume the ciphertext known, C_{known}, was obtained using key $K_{unknown}$ and plaintext $P_{unknown}$ (either of which the attacker is trying to find). The attacker also knows at least C_1 *and* P_1, where C_1 is the output of encrypting P_1 with key $K_{unknown}$. That is, the attacker knows a pair (P_1, C_1). They may also know other pairs (obtained using the same key $K_{unknown}$).

How could an attacker known past plaintext/ciphertext pairs? A simple example is if the plaintext messages were only valid for a limited time, after which they become public. Such as coordinates for a public event to take place. Before the event takes place the coordinates are encrypted and secret. But after the event takes place, while the coordinates were decrypted, the attacker has learnt the value of the coordinates/plaintext (without knowing the key).

Generally, the more pairs of plaintext/ciphertext known, the easiest it is to defeat a cipher.

Chosen Plaintext Attack

- Attacker knows:
  - encryption algorithm
  - ciphertext
  - plaintext message chosen by attacker, together with its corresponding ciphertext generated with the secret key

In a CPA the attacker is able to select plaintexts to be encrypted and obtain their ciphertext (but not knowing the key used in the encryption). In such an attack, the attacker may select plaintext messages that have characteristics that make it easier to break the cipher. Ability to select plaintext and have it encrypted is common for public key ciphers (since the encryption key is public but the decryption key is private), which should be designed to be resistant to such attacks.

Chosen Ciphertext Attack

- Attacker knows:
  - encryption algorithm
  - ciphertext
  - ciphertext chosen by attacker, together with its corresponding decrypted plaintext generated with the secret key
- Attackers aim is to find the secret key (not the plaintext)

In a CCA the attacker chooses a ciphertext, and obtains the corresponding plaintext, in an attempt to discover a secret key. Note in this attack, the aim is to find the secret key. If the attacker has a way to obtain plaintext from a chosen ciphertext, then they could simply intercept ciphertext to find plaintext. A CCA normally involves the attacker tricking a user to decrypt ciphertext and provide the plaintext.

General Measures of Security

Unconditionally Secure Ciphertext does not contained enough information to derive plaintext or key

- One-time pad is only unconditionally secure cipher (but not very practical)

Computationally Secure If:

- cost of breaking cipher exceeds value of encrypted information
- or time required to break cipher exceeds useful lifetime of encrypted information
- Hard to estimate value/lifetime of some information
- Hard to estimate how much effort needed to break cipher

In theory we would like an unconditionally secure cipher. However in practice, we aim for computationally secure. Unfortunately it is difficult to measure if a cipher is computationally secure. For modern ciphers their security is judged based on the known theoretical and practical attacks (e.g. resistant to CCA or not) as well as the metrics in the following.

Common Metrics for Attacks

Time: usually measured as *number of operations*, since real time depends on implementation and computer specifics

- Operations are encrypts or decrypts; ignore other processing tasks
- E.g. worst case brute force of k-bit key takes 2^k (decrypt) operations

Amount of Memory: temporary data needed to be stored during attack

Known information: number of known plaintext/ciphertext values attacker needs to know in advance to perform attack

While time to break the cipher is the metric of interest, it is usually simplified to number of operations. For cryptanalysis, successful attacks should take fewer operations than brute force. That is, an attack that takes more operations the a brute force attack is considered an unsuccessful attack.

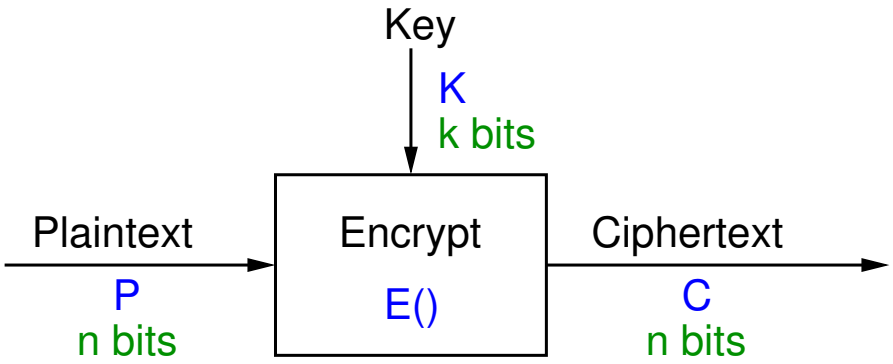
Often attacks requires intermediate values to be stored in memory while performing the attack. The less memory needed, the better the attack.

As seen in the previous classification, known plaintext, chosen plaintext and chosen ciphertext attacks all require the attacker to know additional information. The more information necessary for the attack to be successful, the poorer the attack is. For example, a known plaintext attack that will be successful if 1,000,000 pairs of plaintext/ciphertext are known, is better than a known plaintext attack that requires 2,000,000 pairs.

Contents

Block Cipher with n bit blocks

- Encrypt a block of plaintext as a whole to produce same sized ciphertext
- Typical block sizes are 64 or 128 bits
- Modes of operation used to apply block ciphers to larger plaintexts



Modes of operation are covered in Chapter **??**.

Simple Ideal 2-bit Block Cipher 1

Encryption Cipher 1

P	K0	K1	K2	K3	K4	K5	K6	K7	K8	K9	K10	K11	K12	K13	K14	K15	K16	K17	K18	K19	K20	K21	K22	K23
00	10	10	00	10	11	10	11	00	01	01	00	01	11	01	00	00	10	11	11	01	00	01	10	11
01	00	11	10	11	00	00	10	01	10	00	10	11	10	00	11	01	01	01	00	11	11	10	01	01
10	11	00	11	01	10	01	00	10	11	10	01	10	01	11	01	11	00	10	01	00	10	00	11	00
11	01	01	01	00	01	11	01	11	00	11	11	00	00	10	10	10	11	00	10	10	01	11	00	10

The figure on slide 22 is an example of a 2-bit ideal block cipher. The table shows input plaintext blocks in the left column, different keys in the top row, and the resulting output ciphertext block in the body of the table. To be used for sending a confidential message, both the sender and receiver would know the table (e.g. stored in memory on their devices), or some way to calculate the table) and agree upon the key to use. For a given plaintext block, the sender looks up the key to find the output ciphertext to send. The receiver looks up the receiver ciphertext in the column of the key, and the row determines the plaintext.

Encrypt with Ideal Cipher 1 (exercise)

Encrypt the message *Tokyo* using the above ideal 2-bit block cipher 1 with key K6.

Issues When Applying Block Ciphers

- Encoding/decoding: independent of block cipher, which operate only in binary values
- Mode of operation: typically independent of block cipher, which operate only on a single block
- Repetition of plaintext blocks: undesirable. Make block size larger and use mode of operation that obscures repetition
- Key space: larger block size needed to allow more keys in ideal block cipher
- Implementing an ideal block cipher: how are they generated? can all values be stored?

The following questions will explore some of these issues further.

Simple Ideal 2-bit Block Cipher 2

Encryption Cipher 2

P	K0	K1	K2	K3	K4	K5	K6	K7	K8	K9	K10	K11	K12	K13	K14	K15	K16	K17	K18	K19	K20	K21	K22	K23
00	01	01	00	10	11	00	11	11	01	10	01	00	00	10	01	11	11	01	11	10	00	10	00	10
01	10	11	01	01	11	10	10	01	10	11	11	01	11	00	00	00	01	00	10	01	10	00	11	11
10	11	00	11	00	10	11	01	10	00	01	10	10	10	11	11	01	00	10	00	11	01	01	01	00
11	00	10	10	11	01	01	00	00	11	00	00	11	01	01	10	10	10	11	01	00	11	11	10	01

The figure on slide 25 shows a different 2-bit ideal block cipher. It maps plaintext to ciphertext in a different order than cipher 1.

This example is just used for illustrative purposes. If you had an ideal block cipher that covered every permutation of plaintext values, then only a single cipher is needed.

What is plaintext with key K13, ciphertext 11 with ideal cipher 2? (question)

What is plaintext with key K13, ciphertext 11 with ideal cipher 2?

Decryption also involves a lookup. In the column for key K13, identify the ciphertext 11, and the row indicates the original plaintext 10.

What is plaintext with key K4, ciphertext 11 with ideal cipher 2? (question)

What is plaintext with key K4, ciphertext 11 with ideal cipher 2?

Same cipher, same ciphertext but different key. However in column of K4 there are two values of ciphertext 11. So we cannot determine for sure what was the original plaintext: 00 or 10. This actually is a trick question, since the cipher design is in error. A cipher must be reversible, so decryption is possible. This is an example of a cipher design error that includes an irreversible mapping.

Simple Ideal 2-bit Block Cipher 2 (fixed)

Encryption Cipher 2

P	K0	K1	K2	K3	K4	K5	K6	K7	K8	K9	K10	K11	K12	K13	K14	K15	K16	K17	K18	K19	K20	K21	K22	K23
00	01	01	00	10	11	00	11	11	01	10	01	00	00	10	01	11	11	01	11	10	00	10	00	10
01	10	11	01	01	00	10	10	01	10	11	11	01	11	00	00	00	01	00	10	01	10	00	11	11
10	11	00	11	00	10	11	01	10	00	01	10	10	10	11	11	01	00	10	00	11	01	01	01	00
11	00	10	10	11	01	01	00	00	11	00	00	11	01	01	10	10	10	11	01	00	11	11	10	01

The figure on slide 28 shows the fixed cipher: it is now reversible, and decryption is possible for all values of key and ciphertext.

How many bits are needed to represent the key in cipher 2? (question)

The example 2-bit ideal block cipher 2 (as well as cipher 1) list 24 different keys (or mappings from plaintext to ciphertext). How many bits are needed to represent a key for this cipher?

Firstly, why are 24 keys listed? With a 2-bit block, there are $2^2 = 4$ possible blocks, i.e. 00, 01, 10, and 11. There are $4! = 24$ different ways to arrange those 4 plaintext blocks to produce ciphertext, i.e. 24 permutations of the plaintext blocks. A key is used to select the distinct permutation.

With key length of 1 bit, we can represent $2^1 = 2$ possible keys. With a key length of 2 bits, we can represent $2^2 = 4$ possible keys. With a key length of 3 bits, we can represent $2^3 = 8$ possible keys. With a key length of 4 bits, we can represent $2^4 = 16$ possible keys. With a key length of 5 bits, we can represent $2^5 = 32$ possible keys. That is, a key length of 4 bits is not enough to represent our 24 keys, but a key length of 5 is. Therefore we need a 5-bit key for this ideal 2-bit block cipher.

How to reduce repetition of plaintext blocks? (question)

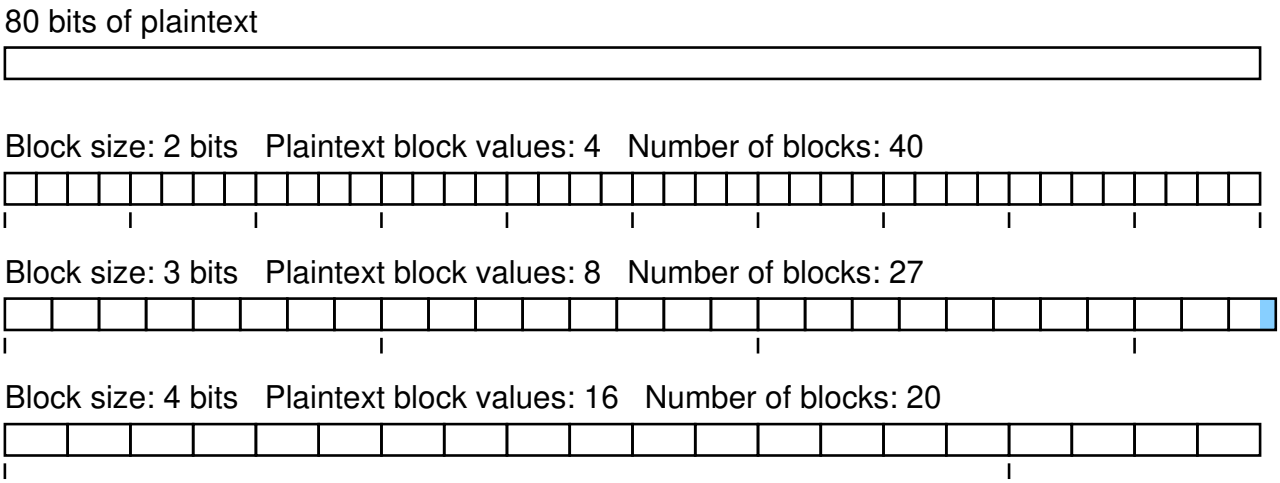
With a 2-bit ideal block cipher, with a long plaintext, many of plaintext blocks will repeat. This is bad for security (see Modes of Operation). What can you change in the design of an ideal block cipher that reduces repetition of plaintext blocks?

Increasing the block size for a block cipher will reduce the change of block repetition. Recall the first example of the 2-bit ideal block cipher encrypting *Tokyo*. The plaintext was 40-bits, resulting in 20 blocks. As there are only $2^2 = 4$ different plaintext values, there will be repetition. On average (if the plaintext was random, which is not likely but it simplifies the analysis), each plaintext value will be repeated $20/4 = 5$ times.

If however a 3-bit ideal block cipher was used, there would be $2^3 = 8$ different plaintext values. There would be 14 blocks (40/3, with the last block having just 1 bit of plaintext). On average, each plaintext value will be repeated $14/8$, which is less than 2 times.

Increasing to a 4-bit ideal block cipher gives 16 different plaintext values, 10 blocks, and a possibility there will be no repetition. Of course if the plaintext is much longer than 40 bits, then repetition is still likely.

Impact of Block Sizes for 80 bit Plaintext



The figure on slide 31 illustrates the impact of different block sizes for an example 80 bit plaintext (whereas the previous example was a 40 bit plaintext).

Note that with a block size of 3 bits, the last block contains 2 bits of plaintext and 1 bit of *padding*. Padding is needed as all blocks must be the same size (since block ciphers operate on fixed sized blocks). There are different schemes for padding, e.g. bit padding, zero padding and PKCS7.

General n-bit Ideal Block Cipher

- n-bit block cipher takes n bit plaintext and produces n bit ciphertext
- 2^n possible different plaintext blocks
- Encryption must be reversible (decryption possible)
- Number of permutations of plaintext (and number of keys) is $2^n!$
- Design trade-offs:
  - Large block size to reduce plaintext repetitions (64-bits is good)
  - Key space large enough to avoid brute force, but small enough to make distribution practical
  - Small block size to simplify implementation

The trade-offs are conflicting, meaning ideal block ciphers are good in theory, but in practice we need a different design approach.

Ideal 64-bit Block Cipher (exercise)

Consider an ideal 64-bit block cipher. How many different different keys are possible? How many bits are needed to store a single key? How much space is required to store the mappings?

Feistel Structure for Block Ciphers

- Ideal block ciphers are not practical
- Feistel proposed applying two or more simple ciphers in sequence so final result is cryptographically stronger than component ciphers
- n-bit block length; k-bit key length; 2^k transformations
- Feistel cipher alternates: substitutions, transpositions (permutations)
- Applies concepts of diffusion and confusion
- Applied in many ciphers today
- Approach:
  - Plaintext split into halves
  - Subkeys (or round keys) generated from key
  - Round function, F, applied to right half
  - Apply substitution on left half using XOR
  - Apply permutation: interchange to halves

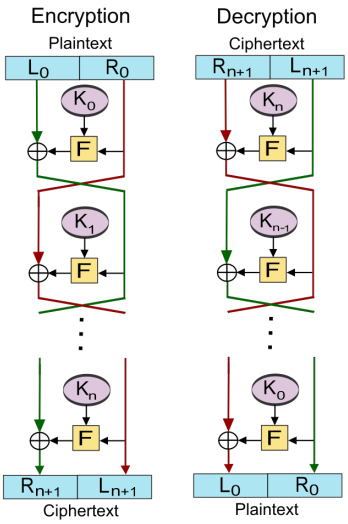
For example, with a 64-bit block cipher, there are 2^{64} possible mappings/keys, meaning the key length is $\log_2(2^{64}) = 64$ bits.

Diffusion and Confusion

- Diffusion
  - Statistical nature of plaintext is reduced in ciphertext
  - E.g. A plaintext letter affects the value of many ciphertext letters
  - How: repeatedly apply permutation (transposition) to data, and then apply function
- Confusion
  - Make relationship between ciphertext and key as complex as possible
  - Even if attacker can find some statistical characteristics of ciphertext, still hard to find key
  - How: apply complex (non-linear) substitution algorithm

Diffusion and confusion are concepts introduced by Claude Shannon. See a summary of Shannon's contributions in telecommunications, digital circuits and cryptography in Chapter **??**.

Feistel Encryption and Decryption



Credit: Amirki, `https://commons.wikimedia.org/wiki/File:Feistel_cipher_diagram_en.svg`, CC BY-SA 3.0

You don't need to know the details of the Feistel structure. Just be aware that it is a design principle used in many block ciphers, including DES.

Using the Feistel Structure

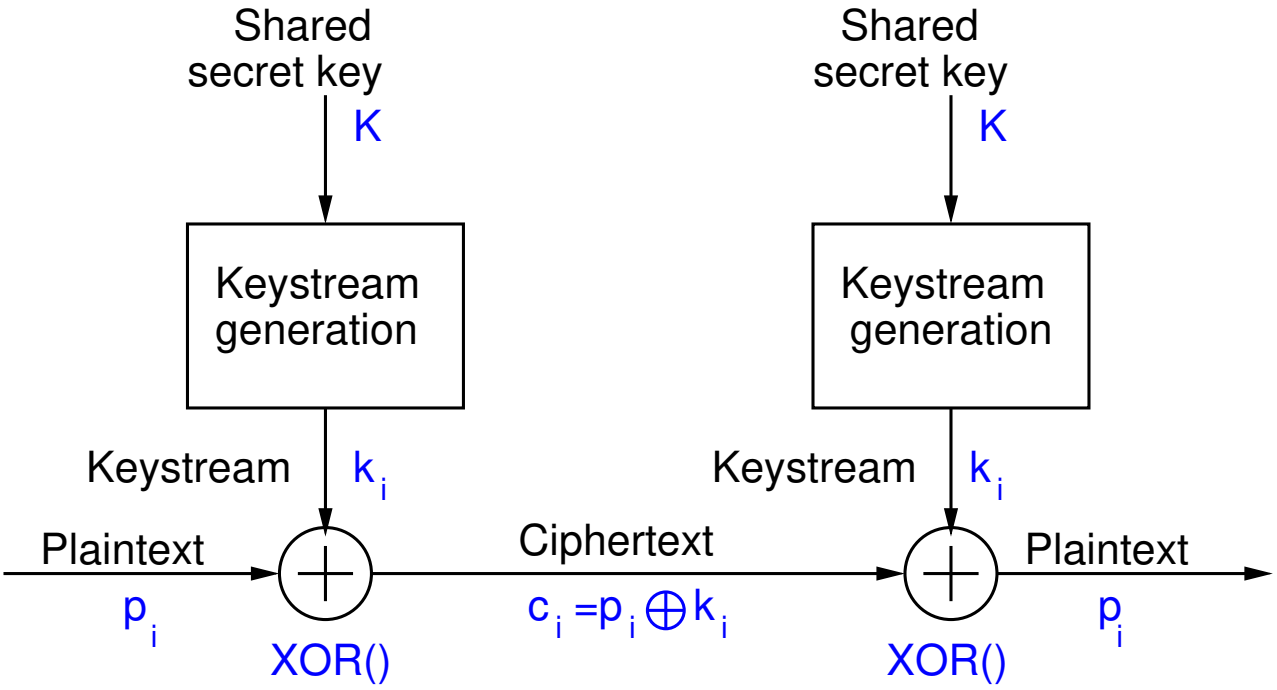
- Exact implementation depends on various design features
  - Block size, e.g. 64, 128 bits: larger values leads to more diffusion
  - Key size, e.g. 128 bits: larger values leads to more confusion, resistance against brute force
  - Number of rounds, e.g. 16 rounds
  - Subkey generation algorithm: should be complex
  - Round function F: should be complex
- Other factors include fast encryption in software and ease of analysis
- Trade-off: security vs performance

Contents

Stream Ciphers

- Encrypts a digital data stream one bit or one byte at a time
- One time pad is example; but practical limitations
- Typical approach for stream cipher:
  - Key (K) used as input to bit-stream generator algorithm
  - Algorithm generates cryptographic bit stream (k_i) used to encrypt plaintext
  - k_i is XORed with each byte of plaintext P_i
  - Users share a key; use it to generate keystream

Stream Cipher Encrypt and Decrypt



The figure on slide 40 illustrates the general operation of a stream cipher encryption and decryption. The sender uses a shared secret key K and an algorithm to generate effectively a random stream of bits. This random stream of bits is XORed with the plaintext bits as needed.

The receiver uses the same key and algorithm, which in turn generates the same random stream of bits. When XORed with the ciphertext, the original plaintext is output.

Key Re-use in Stream Ciphers

- Encrypting two different plaintexts with the same key leads to key re-use attack
  - Attacker intercepts two ciphertexts: $C_1 = P_1 \oplus k_1$ and $C_2 = P_2 \oplus k_1$
  - Properties of XOR: commutative and $A \oplus A = 0$
  - Attacker performs XOR on two ciphertexts
  - $C_1 \oplus C_2 = P_1 \oplus k_1 \oplus P_2 \oplus k_1 = P_1 \oplus P_2$
  - Even without knowing P_1 or P_2, attacker can easily use frequency analysis to discover both
- Solution: Use additional IV that changes for every encryption

When can key re-use attack be successful if IV is used? (question)

If a stream cipher is using a n-bit IV, but the same key, under what conditions is a key re-use attack possible? Assume the IV increments every time an encrypt operation is performed.

Contents

DES and Real Brute Force Attacks

- DES is 64-bit block cipher with 56-bit (effective) key length
- Developed in 1977, recommended standard until 1990's
- Brute force: 2^{56} operations
- Hardware built to perform brute force attack
  - 1998: DeepCrack
  - 2006: COPACABANA

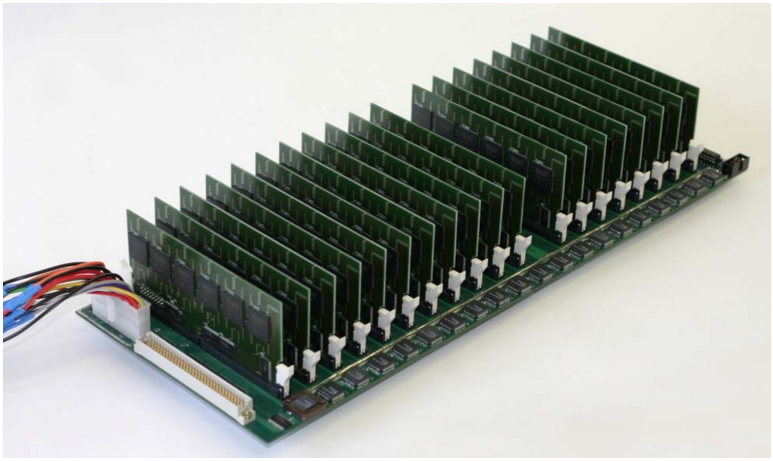
Paul Kocher and DeepCrack

- Developed by EFF
- Cost less than $US250,000
- 80×10^9 keys/sec
- Solved DES challenge in 56 hours
- See `www.cryptography.com` and `www.eff.org`

Credit: Wikimedia, CC0 1.0 Public Domain `https://commons.wikimedia.org/wiki/File:Paul_kocher_deepcrack.jpg`

COPACABANA by SciEngines, 2006

- Joint effort by SciEngines and German universities
- 120 FPGA, 400×10^6 keys/sec/FPGA
- For comparison, a Pentium 4: 2×10^6 keys/sec
- Brute force DES in 8.6 days
- Cost about $US10,000
- See `www.sciengines.com`



Credit: Copyright SciEngines GMBH

Using the above example, we can roughly estimate what it would cost today to brute force DES.

Can We Estimate Cost Today?

- Moore's law: computers double speed every 1.5 years
- Alternative: computers halve in cost every 1.5 years
- $US10,000 to brute force DES in 2006
- Cost has halved about 10 times
- Cost to brute force DES in 2020: $10

A simplification of Moore's law is that computers double their speed every 1.5 years. In practice it is not that simple, but it is a useful rule to estimate the cost of brute force today. It means in 1.5 years time, you could buy a computer that double the speed if a new computer today, and at the same cost. Alternatively, you could buy a lower specced computer, which is the same speed as a new computer today, buy half the cost of today's computer.

Assuming computers halve in cost every 1.5 years, between 2006 and 2020 is 14 years. Over 15 years, there are 10 1.5 year periods, so the cost would halve 10 times. (Again since this is an estimate, let's use 15 years instead of 14). If you half $10,000 10 times, you get $9.76. That is, a $10 computer today can brute force DES in 8.6 days.

As brute force attacks can be parallelised easily, you could spend $100 on 10 computers (or buy a $100 computer) and break DES in less than a day. DES is not secure against a brute force attack (and hasn't been for a long time).

Contents

RIVYERA S3-5000 by SciEngines, 2013

- Rivyera S3 supported up to 128 Xilinx Spartan-3 FPGAs
- Approx $100 per FPGA (XCS5000)
- AES-128 Brute Force
  - 500×10^6 keys per sec
  - 4×10^6 keys per mW
- Biclique Attack
  - 945×10^6 keys per sec
  - 7.3×10^6 keys per mW

Credit: Copyright SciEngines GMBH

FPGA are essentially computer processors programmed for a specific task, in this case, decrypting with AES very fast. For about $12,800 a RIVYERA could decrypt AES-128 at a rate of 500×10^6 keys per second.

A known plaintext attack on AES is called the Biclique attack. The RIVYERA implementation of the Biclique attack could decrypted AES-128 at a rate of 945×10^6 keys per sec, about twice that of a brute force.

Breaking AES-128 in 2020

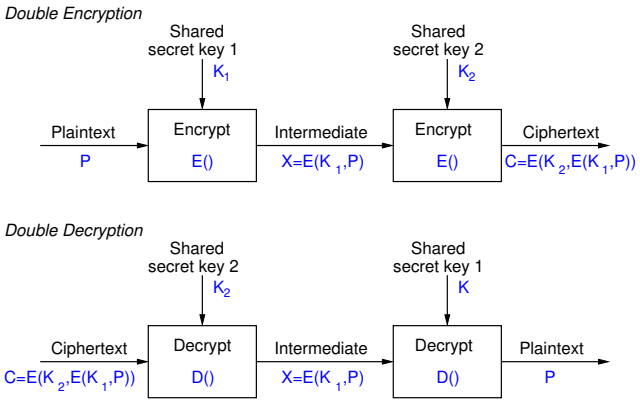
- AES-128 has key space of 2^{128}
- 2013: \$US12,800 for 5×10^8 k/s
- Assume: computers double speed every 1.5 years
- 2020: Increase by $2^5 = 32$; 1.6×10^{10} k/s
  - \$12,800: 6.7×10^{20} years
  - \$12,800,000: 6.7×10^{17} years
  - \$12,800,000,000: 6.7×10^{14} years
- Biclique attack about 2 to 4 times faster, but requires 2^{88} known plaintext/ciphertext pairs
- In 2035, cost \$12,800,000,000 to brute force AES-128 in 670,000,000,000 years

Applying the same logic from analysis of DES brute force and Moore's law (i.e. every 1.5 years halve cost or double speed), we can perform a rough analysis of the cost/time to break AES-128. The numbers (dollars, years) are so large such that even if the approximations are incorrect by a factor of 1,000,000,000 (e.g. reducing 10^{14} years to $100,000$ years, then it is still impossible to break AES-128.

Contents

Double Encryption Concept

- Encrypt plaintext with one key, then encrypt output with another key



- Advantage: doubles the key length
  - Single version of cipher has k-bit key
  - Double version of cipher uses two different k-bit keys
  - Worst case brute force: 2^{2k}
- Advantage: uses an existing cipher
- Disadvantage: doubles the processing time
- Problem: double encryption is subject to *meet-in-the-middle* attack

Double encryption was a (naive) option for extending the key length of DES. It effectively would double the key length from 56 bits to 112 bits. A new cipher would not have to be designed or analysed, and existing software/hardware implementations could be used.

But a meet-in-the-middle attack makes Double-DES (or double encryption on any block cipher) insecure.

Meet-in-the-Middle Attack

- Double Encryption where key K is k-bits: $C = \mathrm{E}(K_2, \mathrm{E}(K_1, P))$
- Say $X = \mathrm{E}(K_1, P) = \mathrm{D}(K_2, C)$
- Attacker knows two plaintext, ciphertext pairs (P_a, C_a) and (P_b, C_b)
  1. Encrypt P_a using all 2^k values of K_1 to get multiple values of X
  2. Store results in table and sort by X
  3. Decrypt C_a using all 2^k values of K_2
  4. As each decryption result produced, check against table
  5. If match, check current K_1, K_2 on C_b. If P_b obtained, then accept the keys
- With two known plaintext, ciphertext pairs, probability of successful attack is almost 1
- Encrypt/decrypt operations required: $\approx 2 \times 2^k$ (twice as many as single encryption)

Example 5-bit Block Cipher

	Ciphertext for key, K:							
P	*000*	*001*	*010*	*011*	*100*	*101*	*110*	*111*
00000	00001	10010	01101	01111	11011	10011	10000	11101
00001	10001	01001	11010	10000	01010	11100	10100	01010
00010	01011	10100	11011	01100	00100	10100	00111	00100
00011	01110	10110	01011	00111	10110	11101	11000	00101
00100	00011	00011	00001	11101	11001	10010	11011	01100
00101	10100	10111	01110	00010	01101	00011	01101	00110
00110	10101	11111	00110	10011	00010	10001	10111	10110
00111	01101	10001	10111	00110	11111	01100	11100	10011
01000	01000	11011	10011	01010	01001	10110	10011	11111
01001	10010	11110	10001	10101	01111	00100	00000	01110
01010	01111	00010	10000	10110	11000	01010	00001	00010
01011	11110	01110	00111	01011	11101	11011	01111	10010
01100	11011	10000	01010	00101	01100	00101	01100	00111
01101	11101	00111	10110	01000	01000	10111	10010	11100
01110	11000	01000	10100	00000	11010	01111	11111	01000
01111	01001	11101	01100	00001	00011	01000	01010	01101
10000	00110	11100	01111	01001	01011	11111	00010	11011
10001	11111	01100	10010	10010	00000	11010	11110	00000
10010	10110	10011	11110	01101	10111	01101	10001	10000
10011	00010	00001	11000	11100	10100	00111	00011	10111
10100	10111	01101	11001	11111	10011	00000	00100	00011
10101	01010	01111	00101	00011	00001	01001	10101	01011
10110	00000	00110	10101	11010	00110	01011	01000	11001
10111	00111	11000	01001	11110	10000	00010	01110	10100
11000	00101	01011	00010	10001	11100	10000	11010	10001
11001	11100	00000	11101	10111	10001	01110	00101	11000
11010	11010	11001	01000	01110	01110	11110	01011	01001
11011	01100	11010	11111	11001	10101	00001	10110	00001
11100	11001	01010	00100	00100	00101	11001	00110	10101
11101	10011	10101	00011	10100	00111	00110	11001	01111
11110	00100	00101	11100	11000	10010	11000	11101	11110
11111	10000	00100	00000	11011	11110	10101	01001	11010

The figure on slide 54 shows an example 5-bit block cipher with a 3-bit key. To encrypt, look in the left column to find the row of the plaintext, then look for the column corresponding to the key. The intersection of row and column gives the ciphertext.

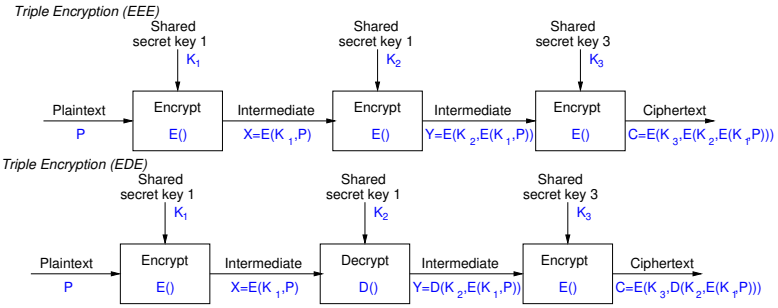
This example block cipher is used in the Meet-in-the-Middle attack exercise.

Meet-in-the-Middle Attack (exercise)

The figure on slide 54 shows an example 5-bit block cipher, referred to as *Bob's Cipher*. A double version of Bob's cipher, called *Double-Bob*, was used by two users to exchange multiple encrypted messages using the same 6-bit secret key. You have obtained the plaintext/ciphertext pairs of two of those messages: $(P_1, C_1) = (01101, 11111)$ and $(P_2, C_2) = (11001, 11011)$. Using a meet-in-the-middle attack, find the secret key.

Triple Encryption Concept

- Different variations:
  - Use 2 keys, e.g. Triple-DES 112 bits
  - Use 3 keys, e.g. Triple-DES 168 bits



- Why E-D-E? To be compatible with single DES:

$$C = \mathrm{E}(K_1, \mathrm{D}(K_1, \mathrm{E}(K_1, P))) = \mathrm{E}(K_1, P)$$

- Problem: 3 times slower than single DES

The figure on slide 56 shows the concept of Triple Encryption, where two different keys are used. This effectively doubles the key strength compared to the original cipher. Another variation (not shown) would be to use three different keys, effectively tripling the key strength.

Note that if you use the same key for each step, then because of the E-D-E approach, this reverts to the original cipher. That is, if you use Triple-DES but use the same key in each step, this reverts to (single) DES. The benefit of this is that you can have an implementation of Triple-DES (which is built on the implementations of DES), and allow the user to choose a key to suit their needs: 1 key for DES, 2 keys for 112-bit security, 3 keys for 168-bit security.

Contents

Cryptanalysis of Triple-DES and AES

Cipher	Method	Key space	Required resources: Time	Memory	Known data
DES	Brute force	2^{56}	2^{56}	-	-
3DES	MITM	2^{168}	2^{111}	2^{56}	2^{2}
3DES	Lucks	2^{168}	2^{113}	2^{88}	2^{32}
AES 128	Biclique	2^{128}	$2^{126.1}$	2^{8}	2^{88}
AES 256	Biclique	2^{256}	$2^{254.4}$	2^{8}	2^{40}

- Known data: chosen pairs of (plaintext, ciphertext)
- Lucks: S. Lucks, Attacking Triple Encryption, in *Fast Software Encryption*, Springer, 1998
- Biclique: Bogdanov, Khovratovich and Rechberger, Biclique Cryptanalysis of the Full AES, in *ASIACRYPT2011*, Springer, 2011